CSE 5243
Homework 2

Soren Goyal, Ohio State University

I. Section 1: Introduction

The main goal of this assignment was to understand the behavior of the kNN classfier using metrics from ROC curves, confusion matrix, etc. The kNN classifier was applied to the two datasets - Iris and Income. More detailed analysis was done for Income Dataset.

A. Datasets

- **Iris Dataset** - The Iris Dataset has four attributes. Each attribute is numeric. It has 150 records, distributed equally among 3 classes - *Setosa, Versicolor and Virginia*.
- **Income Dataset** - The Income Dataset has 15 attributes, which are a mix of nominal, numeric and ordinal data. There are a total of 520 records in the training set and 288 in the test set. There are only two classes. The ratio between the classes is approximately $1 : 3$ with the negative class being larger in number.

B. KNN Algorithm

The prediction of KNN algorithm was broken down into main functions. The first one was `computeDistances(train_data, test_data)` and the second one was `predictLabel(k, distances)`. To classfiy a new records the `predictLabel(k, distances)`

procedure COMPUTEDISTANCES($train\_data, test\_data$)
  for i in test_data **do**
    distance[] = zeros()
    for j in train_data **do**
      distance[j] ← computeDistance(r,s)
    end for
    distances[i,j] ← sort(distance, order = increasing)
    labels[i,j] ← getLabels(distance) ▷ *Gives the training data labels of the classes sorted according to distance from test record*
    return distances, labels
  end for
end procedure

The `computeDistances()` function computes the distance of each of the test record from each of the training data record. The distances are sorted and stored for use by `predictLabel(k, distances)`. `predictLabel(k, distances)` takes the number of nearest neighbors (k) as a parameter and predicts the label. The detailed defintions of the functions are given in fig. ?? and fig. ??.

procedure PREDICTLABEL($k, distances, labels$)
  weight[] = 0,0
  for i in 1:nrow(distances) **do**
    for i in 1:k **do** weight[labels[i,j]] = weight[labels[i,j]] + 1/distances[i,j]
    end for
    prob[i] = weight[2]/sum(weights)
  end for
end procedure

The reason for separating out the distance computation was the predictions were to be made for different values of k. Distance computation takes $O(mn \log n)$ time, where m is the number of records in test dataset, n is the number of records in training dataset. While prediction takes only $O(mk)$ time, which is considereably less than $O(mn \log n)$.
To classify the test records, the KNN algorithm was used. A number of variations were tried in each of the algorithms. This section explains each of the variations and discusses the pros and cons of each.

1) Variation in Preprocessing

- **Simple Scaling** - The nominal attributes are not touched. Numeric and ordinal attributes are scaled and centered. The resultant standard deviation was 1 and mean was 0.
- Simple Scaling with Binning - Apart from scaling and centering the numeric and ordinal attributes the nominal attriutes were also modified. Many of the nominal attributes were very skewed. For e.g - In *Native Country* in Income dataset, out of 26 categories, 12 categories had only one record each. In such cases the smalled categories were grouped together. Also *Education_cat* was converted to nominal type by grouping a few educational levels what had very few records.

2) Variations in Proximity Metric

- **Euclidean Distance-Simple Dissimilarity** - For numeric attributes euclidean distances were computed, while for nominal attributes simple dissimilarity was computed. These were combined by taking a weighted average.
- **Cosine Similarity-Simple Dissimilarity** - Nominal atributes were treated the same way as the previous type. But for numeric attributes the cosine similarity was computed. Finally, they were combined in the same way as the previous method i.e by taking a weighted average.

3) Algorithm for Voting

Let N represent the set of k nearest neighbors. Let $N`(1) \subseteq N$ be set of neighbors belonging to class 1 and $N`(2)$ be the set of neighbors belonging to the other class. **Inverse Distance**

Weighted Voting: Once the k nearest neighbours have been determined, the probablity of a point belonging to a class is as follows -

$$P(class = i) = \frac{\sum_{r \in N(i)} \frac{1}{distance(r)}}{\sum_{r \in N} \frac{1}{distance(r)}}$$

Finally, the class was assigned based on a threhold that can be set to get the target true positive rate or false psoitive rate.

II. Section 2: Evaluation of performance

The variations mentioned in the previous section were tried and the analysis of the perfromance of KNN classifier is presented in this section.

A. Classifier Performance for Iris Dataset

Classifier 1

- Preprocessing - Simple Scaling
- Proximity Metric - Euclidean Distance
- Voting - Inverse Distance Weighted Voting

The confusion matrix for k = 5 is given in Fig: ? From the

Fig. 1. Confusion Matrix k = 5 (Datset:Iris Classifier:1)

		Predicted Label		
		Setosa	Versicolor	Virginica
Actual Label	Setosa	20	0	0
	Versicolor	1	21	8
	Virginica	0	3	17

analysis in the previous assignment, we know that the setosa class was distinctly separated from the other two classes. The effect of this can be seen in the confusion matrix. The KNN classifier was able to classify all instances of setosa correctly. Also, there was only 1 false positive for setosa. On the other hand, versicolor and virginica had significant overlap. The algorithm did a good job on them too. Although, roughly 33% of Versicolor were falsely classified as Virginica. Overall the accuracy of this classfier was 82%

Classifier 2

- Preprocessing - Simple Scaling (same as before)
- Proximity Metric - Cosine Similarity
- Voting - Inverse Distance Weighted Voting (same as before)

The confusion matrix for $k = 5$ is given in Fig: ??. This classifier performed better than the previous one. The accuracy increased to 87%, by simply by using cosine similarity instead of euclidean similarity. Here to setosa was much more easily distinguised as compared to the other two classes, although the false positives in setosa increased increased by small amount.

Fig. 2. Confusion Matrix k = 5 (Datset:Iris Classifier:2)

		Predicted Label		
		Setosa	Versicolor	Virginica
Actual Label	Setosa	20	0	0
	Versicolor	2	24	4
	Virginica	0	3	17

B. Classifier Performance for Income Dataset

For binary income dataset, the classifier had to be optimized over two parameters had to be optimized - the number of nearest neighbours k and threshold for making the prediction. To optimum k for the classifier would be the k for which the "Area Under the Roc Curve" is maximized. To find the optimum k, the roc graphs were computed over the test set for k's ranging from 1 to 100. It was expected that the area would increase with k, peak and then plateau out. The k at the peak would then be optimum.
For a picking the optmium threshold, we maximized the accuracy of the classifier. To determine the highest accuracy, the threshold was varied keeping the k was fixed. The threshold for which the accuracy is maximum was taken to be the optimum threshold.

Classifier 1

- Preprocessing - Simple Scaling
- Proximity Metric - Euclidean Distance
- Voting - Inverse Distance Weighted Voting

For this classifier the Area under ROC reaches maximum at $k = 38$. In Fig. ?? four ROC curves have been plotted. The ROC curve for $k = 38$ has the most area underneath it, although it is surpassed at few points by other ROC curves. The accuracy for $k = 38$ is plotted as a function of $threshold$. It increase monotonically as $threshold$ increases and is maximum for $threshold = 1$ (point not on the curve). This is beacuse of the highly skewed class ratio, the highest accuracy is obtained when all records are classified as negative. This is clearly not a $threshold$ to pick. However, if we give more weightage to true positives, the Accuracy Vs Threshold Curve might for values less than one, giving a more reasonable classifier.

Classifier 2

- Preprocessing - Simple Scaling
- Proximity Metric - Cosine Similarity
- Voting - Inverse Distance Weighted Voting

The Iris dataset should a good improvement when the proximity metric was changed from eulidean distance to cosine similarity. However, the same is not observed in income dataset. For this classifier the "area under the curve" reaches its maximum at k=38. Fig. ?? contains the plot for the AUC. Four different ROCs have been plotted in Fig. ??. The trend in AUC and ROC is similar to what was observed in previous

Fig. 3. Plot of Area Under the Roc curve for different values of k for Income Classifier 1. Maxima is achieved at k = 38

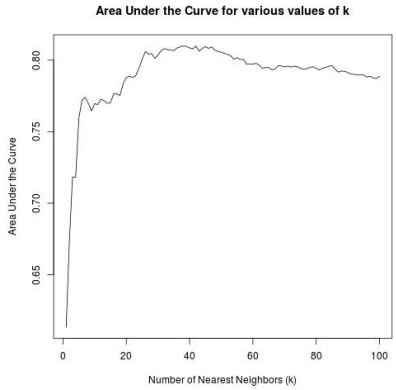
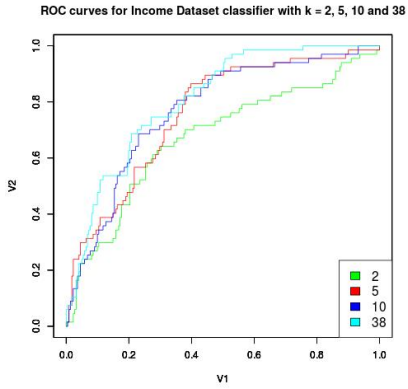


Fig. 4. ROC curves for different values of k for Income Classifier 1



classifier. The best accuracy too is obtained for $threshold = 1$. The false positive rate increase way to fast in comparison to the true positive rate.

Classifier 3 - Best Performance

- Preprocessing - Simple Scaling with Binning
- Proximity Metric - Cosine Similarity
- Voting - Inverse Distance Weighted Voting

Based on the previous performance of the previous two classfiers, it was felt that the attributes should be preprocessed

Fig. 5. Plot of Accuracy Vs Threshold Values for k = 38

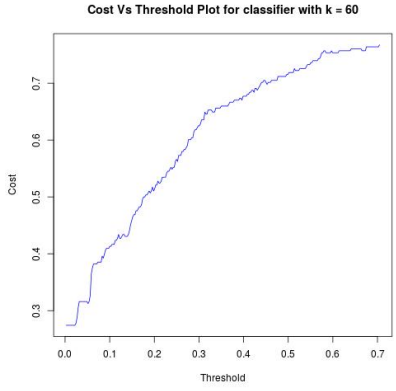


Fig. 6. Plot of Area Under the Roc curve for different values of k for Income Classifier 2. Maxima is achieved at k = 36

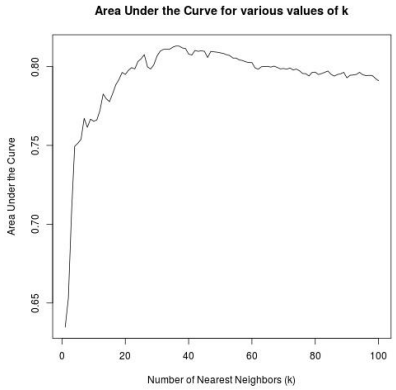
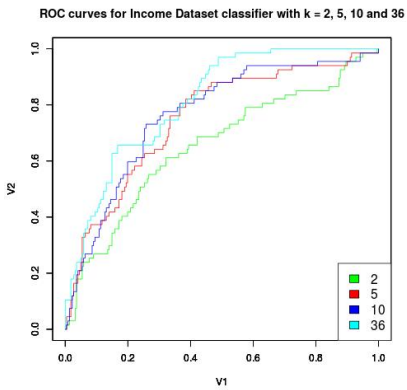


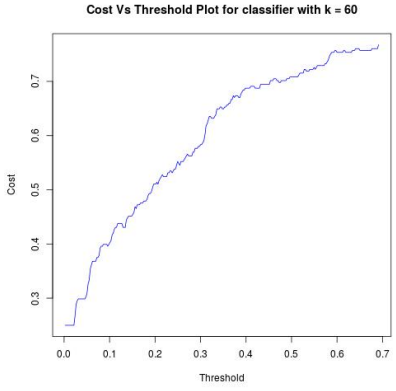
Fig. 7. ROC curves for different values of k for Income Classifier 2



such that the similar record have similar values. In that spirit, many categories were merged. For e.g - There were 26 native countries, with 12 having only one record. Each of these records belonged to the same class. The records would be 1 unit apart if simple dissimilarity is computed. But for this classifier all these records were bunched together in a single category and hence bringing them closer.

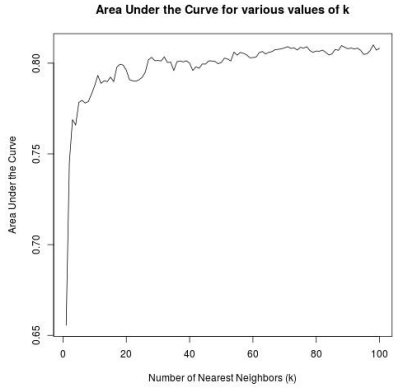
These preprocessing did have a small postive effect. The maximum area under the curve incrased by 5% (Fig. ??). The ROC for optimium k also reaches a saturation at lower FPR

Fig. 8. Plot of Accuracy Vs Threshold Values for k = 36



as compared to Classfier 1 or 2. However, the the optimium $threshold$ is still 1.0.

Fig. 9. Plot of Area Under the Roc curve for different values of k for Income Classifier 3. Accuracy keeps increaing beyond hundred



C. Answer to Select Homework Questions

A,B) The Precision, Recall and F-Measure are plotted in Fig. ??

D) There was effect of proximity measure. It was more pronuced on Iris than Income dataset.

E) To understand the effect of k on the classifier the AUC can be observed. As mentioned earlier in this report the classfier becomes better for a while, then it peaks and then plateaus. (Fig. ??)

F) The third varion in Section. ?? was added becuase the algorithm was not working well with data.

III. Section 3: Comparision with Off-the-Shelf kNN

The off-the-shelf knn classfier used was the one implemented in the R package *'class'* written by Brian Ripley. The key settings used for this classification were -

- Preprocessing - Simple Scaling with Binning(same as classifier 3)
- Proximity Metric - (internal)

Fig. 10. ROC curves for different values of k for Income Classifier 3

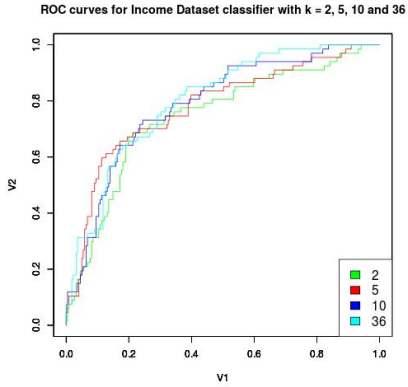


Fig. 11. Plot of Accuracy Vs Threshold Values for k = 40

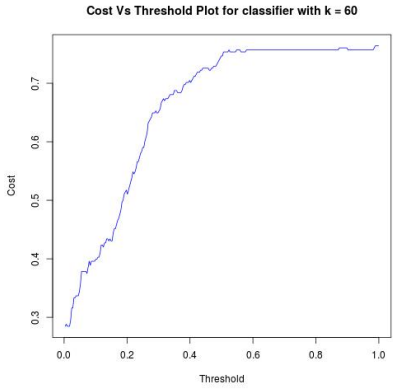
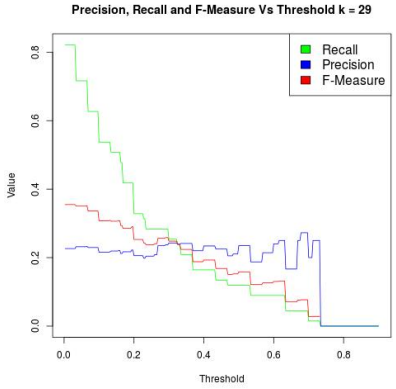


Fig. 12. Precision, Recall and F-Measure for Classifier 3 of Income Data test set



- Voting - (internal)

The classifier performed significantly better than any of our classifiers. The AUC peaked at $k = 29$. The ROC curve at $k = 29$ also rose sharply and then saturated at $FPR = 0.6$. The AUC was greater than either of out classfiers. Finally the accuracy also rose quickly with threshold and reached a saturation at 0.8.

Fig. 13. Plot of Area Under the Roc curve for different values of k for Income dataset for off-the-shelf-knn. Maxima is achieved at k = 29

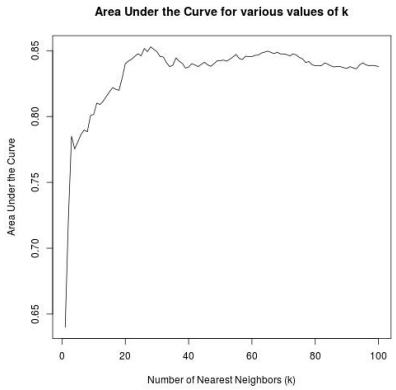


Fig. 14. ROC curves for different values of k for Income Classifier 2

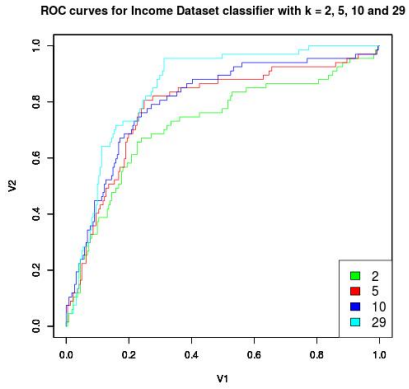


Fig. 15. Plot of Accuracy Vs Threshold Values for k = 29